CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT
Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center

Kreatif Industri Nusantara

Penulis:
Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:
M. Yusril Helmi Setyawan

Penyunting:
Syafrial Fachrie Pane
Khaera Tunnisa
Diana Asri Wijayanti

Desain sampul dan Tata letak:
Deza Martha Akbar

Penerbit:
Kreatif Industri Nusantara

Redaksi:
Jl. Ligar Nyawang No. 2
Bandung 40191
Tel. 022 2045-8529
Email : awangga@kreatif.co.id

Distributor:
Informatics Research Center
Jl. Sariasih No. 54
Bandung 40151
Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

'Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu harus
sanggup menahan
perihnya Kebodohan.'
Imam Syafi'i

CONTRIBUTORS

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

CONTENTS IN BRIEF

DAFTAR ISI

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

Bandung, Jawa Barat
Februari, 2019

ACKNOWLEDGMENTS

Terima kasih atas semua masukan dari para mahasiswa agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk team IRC yang telah fokus untuk belajar dan memahami bagaimana buku ini mendampingi proses Intership.

R. M. A.

ACRONYMS

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association

GLOSSARY

git	Merupakan manajemen sumber kode yang dibuat oleh linus torvald.
bash	Merupakan bahasa sistem operasi berbasiskan *NIX.
linux	Sistem operasi berbasis sumber kode terbuka yang dibuat oleh Linus Torvald

SYMBOLS

A Amplitude

& Propositional logic symbol

a Filter Coefficient

$\mathcal{B}$ Number of Beats

INTRODUCTION

ROLLY MAULANA AWANGGA, S.T., M.T.

Informatics Research Center
Bandung, Jawa Barat, Indonesia

Pada era disruptif saat ini. git merupakan sebuah kebutuhan dalam sebuah organisasi pengembangan perangkat lunak. Buku ini diharapkan bisa menjadi penghantar para programmer, analis, IT Operation dan Project Manajer. Dalam melakukan implementasi git pada diri dan organisasinya.

Rumusnya cuman sebagai contoh aja biar keren[1].

$$ABC\mathcal{D}\mathcal{E}\mathcal{F}\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

BAB 1

CHAPTER 1

BAB 2

CHAPTER 2

BAB 3

CHAPTER 3

BAB 4

CHAPTER 4

BAB 5

CHAPTER 5

BAB 6

CHAPTER 6

BAB 7

CHAPTER 7

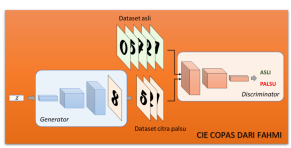
BAB 8

CHAPTER 8

8.1 1164013 - Ikrima Ningrum

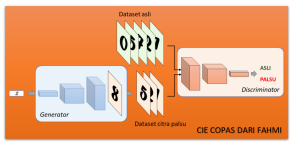
8.1.1 Soal Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa itu generator dengan perumpamaan anda sebagai mahasiswa sebagai generatornya.
   Tugas Generator sekarang sedang dibuat untuk membuat koleksi gambar palsu, yang saat ini dilihat oleh Diskriminator. Diskriminator tidak dapat membedakan antara yang asli dan yang palsu. Untuk ilustrasi, lihat gambar berikut:



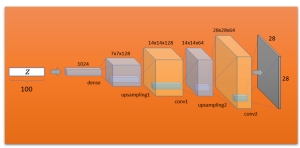
Gambar 8.1 Teori 1

2. Jelaskan dengan ilustrasi gambar sendiri apa itu diskriminator dengan perumpamaan dosen anda sebagai diskriminatornya.
   Diskriminator adalah CNN yang menerima input gambar yang dimiliki dan menghasilkan angka biner yang meminta input gambar, lalu menghasilkan gambar dari dataset asli atau menghasilkan gambar palsu. Untuk ilustrasi, lihat gambar berikut:



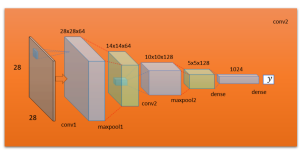
Gambar 8.2 Teori 2

3. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur generator dibuat.
   Aksitektur generator dibuat bisa dijelaskan pada gambar berikut :



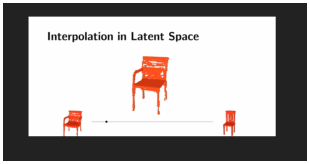
Gambar 8.3 Teori 3

4. Jelaskan dengan ilustrasi gambar sendiri bagaimana arsitektur diskriminator dibuat.
   Aksitektur diskriminator dibuat bisa dijelaskan pada gambar berikut :



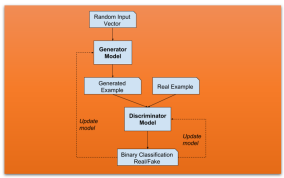
Gambar 8.4 Teori 4

5. Jelaskan dengan ilustrasi gambar apa itu latent space.
   Latent space dijelaskan pada gambar berikut :



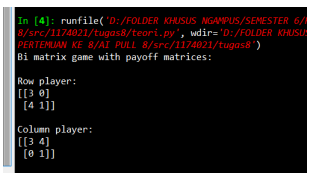
Gambar 8.5 Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu adversarial play.
   Adversarial play dijelaskan pada gambar berikut :



Gambar 8.6 Teori 6

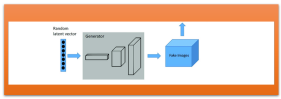
7. Jelaskan dengan ilustrasi gambar apa itu Nash equilibrium.
   Nash equilibrium adalah Teori permainan adalah studi tentang interaksi strategis antara agen rasional. Sederhananya itu berarti itu adalah studi interaksi ketika pihak-pihak yang terlibat mencoba dan melakukan yang terbaik dari sudut pandang mereka, detailnya dapat dijelaskan pada gambar berikut :



Gambar 8.7 Teori 7

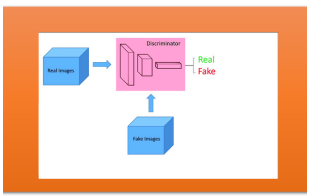
8. Sebutkan dan jelaskan contoh-contoh implementasi dari GAN.
   Menurut saya implementasi 3DGAN yaitu pada MAPS dan juga IKEA. Karena pada maps dan juga ikea sudah menerapkan bentuk 3 dimensi yang bisa lebih menarik perhatikan pengguna.

9. Berikan contoh dengan penjelasan kode program beserta gambar arsitektur untuk membuat generator(neural network) dengan sebuah input layer, tiga hidden layer(dense layer), dan satu output layer(reshape layer).
   Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



Gambar 8.8 Teori 9

10. Berikan contoh dengan ilustrasi dari arsitektur dikriminator dengan sebuath input layer, 3 buah hidden layer, dan satu output layer.
    Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini :



Gambar 8.9 Teori 10

11. Jelaskan bagaimana kaitan output dan input antara generator dan diskriminator tersebut. Jelaskan kenapa inputan dan outputan seperti itu.
    Gambar dari Generator yang berhasil di deteksi oleh Diskriminator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminator, lalu, Diskriminator tidak bisa membedakan fake dan real.

12. Jelaskan apa perbedaan antara Kullback-Leibler divergence (KL divergence)/relativeentrop
    Jensen-Shannon(JS) divergence / information radius(iRaD) / total divergence to the average dalam mengukur kualitas dari model.
    Perbedaan nya yaitu memiliki model dari rumus yang berbeda-beda sehingga mempengaruhi hasil train dan test

13. Jelaskan apa itu fungsi objektif yang berfungsi untuk mengukur kesamaan antara gambar yang dibuat dengan yang asli.
    Ukuran penting untuk menilai kualitas model. lalu kemudian akan melihat di keseimbangan Nash.

14. Jelaskan apa itu scoring algoritma selain mean square error atau cross entropy seperti The Inception Score dan The Frechet Inception distance.
    The inception score adalah algoritma penilaian yang paling banyak digunakan untuk GAN. The Frechet Inception distance adalah Untuk mengatasi berbagai kekurangan Skor awal

15. Jelaskan kelebihan dan kekurangan GAN.
    Kelebihan : GAN dapat memvisualiasikan bentuk model menjadi plot. Kemudian pada Kelemahan : model susah untuk implementasikan yang membuat data training menjadi lemah.

8.1.2 Praktek Program

1. Soal 1
   Konvolusi 3D. Konvolusi 3D menerapkan filter 3 dimensi ke kumpulan data dan filter 3 arah (x, y, z) untuk menghitung representasi fitur tingkat rendah. Bentuk outputnya adalah ruang volume 3 seperti kubus atau berbentuk kubus. 3D sangat membantu dalam pendeteksian peristiwa dalam video, gambar medis 3D, dll. Generative Adversarial Network adalah arsitektur jaringan saraf tiruan yang dimaksudkan untuk membuat atau membuat data yang benar-benar baru, dari nol hingga tidak ada sama sekali. Melihat target utama GAN adalah data gambar. Singkatnya, jaringan GAN berfungsi untuk memberikan gambar baru berdasarkan koleksi gambar yang telah ada sebelumnya selama proses training.

2. Soal 2

```
def build_generator():
    """
    Create a Generator Model with hyperparameters values defined as follows
    """
    z_size = 200
    gen_filters = [512, 256, 128, 64, 1]
    gen_kernel_sizes = [4, 4, 4, 4, 4]
    gen_strides = [1, 2, 2, 2, 2]
    gen_input_shape = (1, 1, 1, z_size)
    gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
    gen_convolutional_blocks = 5

    input_layer = Input(shape=gen_input_shape)

    # First 3D transpose convolution(or 3D deconvolution) block
    a = Deconv3D(filters=gen_filters[0],
                 kernel_size=gen_kernel_sizes[0],
                 strides=gen_strides[0])(input_layer)
    a = BatchNormalization()(a, training=True)
    a = Activation(activation='relu')(a)

    # Next 4 3D transpose convolution(or 3D deconvolution) blocks
    for i in range(gen_convolutional_blocks - 1):
        a = Deconv3D(filters=gen_filters[i + 1],
                     kernel_size=gen_kernel_sizes[i + 1],
                     strides=gen_strides[i + 1], padding='same')(a)
        a = BatchNormalization()(a, training=True)
        a = Activation(activation=gen_activations[i + 1])(a)

```

```
    gen_model = Model(inputs=[input_layer], outputs=[a])
    return gen_model
```

Kode di atas akan melakukan create generator ialah gloss, Bentuk jaringan Generator dapat dilihat berkebalikan dengan struktur jaringan saraf pada umumnya. Generator biasanya menerima input sebuah vektor z, yang kemudian mengubahnya menjadi sebuah output 3D atau 3 dimensi.

3. Soal 3

```
def build_discriminator():
    """
    Create a Discriminator Model using hyperparameters values
    defined as follows
    """

    dis_input_shape = (64, 64, 64, 1)
    dis_filters = [64, 128, 256, 512, 1]
    dis_kernel_sizes = [4, 4, 4, 4, 4]
    dis_strides = [2, 2, 2, 2, 1]
    dis_paddings = ['same', 'same', 'same', 'same', 'valid']
    dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
    dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
                       'leaky_relu', 'sigmoid']
    dis_convolutional_blocks = 5

    dis_input_layer = Input(shape=dis_input_shape)

    # The first 3D Convolutional block
    a = Conv3D(filters=dis_filters[0],
               kernel_size=dis_kernel_sizes[0],
               strides=dis_strides[0],
               padding=dis_paddings[0])(dis_input_layer)
    # a = BatchNormalization()(a, training=True)
    a = LeakyReLU(dis_alphas[0])(a)

    # Next 4 3D Convolutional Blocks
    for i in range(dis_convolutional_blocks - 1):
        a = Conv3D(filters=dis_filters[i + 1],
                   kernel_size=dis_kernel_sizes[i + 1],
                   strides=dis_strides[i + 1],
                   padding=dis_paddings[i + 1])(a)
        a = BatchNormalization()(a, training=True)
        if dis_activations[i + 1] == 'leaky_relu':
            a = LeakyReLU(dis_alphas[i + 1])(a)
        elif dis_activations[i + 1] == 'sigmoid':
            a = Activation(activation='sigmoid')(a)

    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
    return dis_model
```

Diskrimanator adalah d_loss, Jaringan Discriminator merupakan jaringan klasifikasi biner yang menerima input gambar tiga dimensi dan mengeluarkan klasifikasi menyatakan input gambar adalah gambar asli dari dataset atau merupakan

gambar buatan Generator. Diskriminator dilatih dengan dataset yang diambil dari Generator, lalu di training untuk membedakan keduanya. Gambar dari Generator yang berhasil di deteksi oleh Diskriminator sebagai gambar fake, akan dikembalikan dengan feedback pke generator. Kini Generator bertugas untuk bisa membuat sekumpulan gambar palsu, yang nantinya dapat dilihat oleh Diskriminator, lalu, Diskriminator tidak bisa membedakan fake dan real.

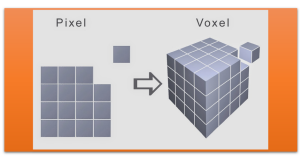
4. Soal 4
   Proses training 3D GAN yaitu dengan melakukan epoch sebanyak yang ditentukan.

5. Soal 5

   - Clone github
   - Download dataset
   - Buat folder baru logs dan results

6. Soal 6
   Dataset yang digunakan yaitu 3DShapeNets yang berisi model model bentuk benda dll, folder train berisi train dan folder test berisi data testing. dan semua data tersebut di simpan didalam folder volumetric_data.

7. Soal 7
   Volume pixel atau voxel adalah titik dalam ruang tiga dimensi. Sebuah voxel mendefinisikan posisi dengan tiga koordinat dalam arah x, y, dan z. Sebuah voxel adalah unit dasar untuk mewakili gambar 3D. Untuk gambar nya ialah sebagai berikut :



Gambar 8.10 Praktek 7

8. Soal 8

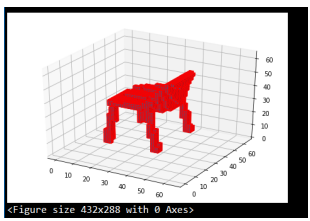
```
# -*- coding: utf-8 -*-
"""
Created on Sun May 10 18:10:55 2020

@author: Ikrimaa


"""
```

```
9
10 # In[]
11
12 import scipy.io as io
13 voxels = io.loadmat("data/3DShapeNets/volumetric_data/chair/30/
       test/chair_000000000_1.mat")['instance']
14 #%%
15 import numpy as np
16 voxels = np.pad(voxels, (1, 1), 'constant', constant_values=(0,
       0))
17 #%%
18 import scipy.ndimage as nd
19 voxels = nd.zoom(voxels, (2, 2, 2), mode='constant', order=0)
20 #%%
21 import matplotlib.pyplot as plt
22 from mpl_toolkits.mplot3d import Axes3D
23 fig = plt.figure()
24 ax = Axes3D(fig)
25 ax.voxels(voxels, edgecolor="red")
26
27 plt.show()
28 plt.savefig('data')
```

Kode di atas befungsi untuk visualisasidataset dalam tampilan plot. langkah-langkah seperti ini : import library, load data file .mat dan lakukan read memakai matplotlib, Hasilnya adalah sebagai berikut :



Gambar 8.11 Praktek 8

9. Soal 9

```
1 #%% soal9
2
3 def build_generator():
4     """
5     Create a Generator Model with hyperparameters values defined
      as follows
6     """
7     z_size = 200
8     gen_filters = [512, 256, 128, 64, 1]
9     gen_kernel_sizes = [4, 4, 4, 4, 4]
10    gen_strides = [1, 2, 2, 2, 2]
11    gen_input_shape = (1, 1, 1, z_size)
12    gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
```

```
    gen_convolutional_blocks = 5

    input_layer = Input(shape=gen_input_shape)

    # First 3D transpose convolution(or 3D deconvolution) block
    a = Deconv3D(filters=gen_filters[0],
                 kernel_size=gen_kernel_sizes[0],
                 strides=gen_strides[0])(input_layer)
    a = BatchNormalization()(a, training=True)
    a = Activation(activation='relu')(a)

    # Next 4 3D transpose convolution(or 3D deconvolution) blocks
    for i in range(gen_convolutional_blocks - 1):
        a = Deconv3D(filters=gen_filters[i + 1],
                     kernel_size=gen_kernel_sizes[i + 1],
                     strides=gen_strides[i + 1], padding='same')(
    a)
        a = BatchNormalization()(a, training=True)
        a = Activation(activation=gen_activations[i + 1])(a)

    gen_model = Model(inputs=[input_layer], outputs=[a])
    return gen_model
```

Kode di atas befungsi untuk membuat generator yaitu dengan ketentukan gen sebagai variabel dan membuat fungsi atau variabel genmodel lalu dilakukan return.

10. Soal 10

```
#%% soal10


def build_discriminator():
    """
    Create a Discriminator Model using hyperparameters values
    defined as follows
    """

    dis_input_shape = (64, 64, 64, 1)
    dis_filters = [64, 128, 256, 512, 1]
    dis_kernel_sizes = [4, 4, 4, 4, 4]
    dis_strides = [2, 2, 2, 2, 1]
    dis_paddings = ['same', 'same', 'same', 'same', 'valid']
    dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
    dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
                       'leaky_relu', 'sigmoid']
    dis_convolutional_blocks = 5

    dis_input_layer = Input(shape=dis_input_shape)

    # The first 3D Convolutional block
    a = Conv3D(filters=dis_filters[0],
               kernel_size=dis_kernel_sizes[0],
               strides=dis_strides[0],
               padding=dis_paddings[0])(dis_input_layer)
```

```
    # a = BatchNormalization()(a, training=True)
    a = LeakyReLU(dis_alphas[0])(a)

    # Next 4 3D Convolutional Blocks
    for i in range(dis_convolutional_blocks - 1):
        a = Conv3D(filters=dis_filters[i + 1],
                   kernel_size=dis_kernel_sizes[i + 1],
                   strides=dis_strides[i + 1],
                   padding=dis_paddings[i + 1])(a)
        a = BatchNormalization()(a, training=True)
        if dis_activations[i + 1] == 'leaky_relu':
            a = LeakyReLU(dis_alphas[i + 1])(a)
        elif dis_activations[i + 1] == 'sigmoid':
            a = Activation(activation='sigmoid')(a)

    dis_model = Model(inputs=[dis_input_layer], outputs=[a])
    return dis_model
```

Kode di atas befungsi untuk membangun diskriminator berfungsi untuk mendefenisikan seluruh gambar yang sudah di load generator sebagai gambar fake dan real.

11. Soal 11
    Jika interpreter python menjalankan if name == main sebagai program utama, itu ialah menetapkan variabel name untuk memiliki nilai main. Jika file ini sedang di impor dari modul lain, name akan ditetapkan ke nama modul. Nama modul tersedia sebagai nilai untuk name variabel global.

12. Soal 12

```
#%% soal 12
    object_name = "airplane"
    data_dir = "data/3DShapeNets/volumetric_data/" \
               "{}/30/train/*.mat".format(object_name)
    gen_learning_rate = 0.0025
    dis_learning_rate = 10e-5
    beta = 0.5
    batch_size = 1
    z_size = 200
    epochs = 1
    MODE = "train"
```

Kode di atas befungsi untuk melakukan load dataset dengan ketentuan data yang hanya dalam folder chair pada data train.

13. Soal 13

```
#%% soal 13
    """
    Create models
    """
    gen_optimizer = Adam(lr=gen_learning_rate, beta_1=beta)
```

```
    dis_optimizer = Adam(lr=dis_learning_rate, beta_1=beta)

    discriminator = build_discriminator()
    discriminator.compile(loss='binary_crossentropy', optimizer=
    dis_optimizer)

    generator = build_generator()
    generator.compile(loss='binary_crossentropy', optimizer=
    gen_optimizer)
```

Kode di atas menggunakan Adam sebagai algoritma pengoptimalan dan binary_crossentropy sebagai kerugian loss.

14. Soal 14

```
#%% soal14
    discriminator.trainable = False

    input_layer = Input(shape=(1, 1, 1, z_size))
    generated_volumes = generator(input_layer)
    validity = discriminator(generated_volumes)
    adversarial_model = Model(inputs=[input_layer], outputs=[
    validity])
    adversarial_model.compile(loss='binary_crossentropy',
    optimizer=gen_optimizer)
```

Kode di atas artinya ialah kita memasukkan random vector kedalam generator model lalu membagi 2 yaitu generated example dan real example, dan meneruskan ke diskriminator model sebagai real atau fake

15. Soal 15

```
#%% soal15
    print("Loading data...")
    volumes = get3DImages(data_dir=data_dir)
    volumes = volumes[..., np.newaxis].astype(np.float)
    print("Data loaded...")
```

Kode di atas befungsi untuk melakukan load data pada dataset.

16. Soal 16

```
#%% soal16
    tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
    ()))
    tensorboard.set_model(generator)
    tensorboard.set_model(discriminator)
```

Kode di atas berfungsi untuk membuat tensorboard yang nantinya bisa diakses melalui localhost.

17. Soal 17

```
#%% soal17
    labels_real = np.reshape(np.ones((batch_size,)), (-1, 1, 1, 1, 1))
    labels_fake = np.reshape(np.zeros((batch_size,)), (-1, 1, 1, 1, 1))
```

Kode di atas befungsi untuk melakukan reshape agar shape yang dihasilkan tidak terlalu besar. Dengan membuat variabel real dan fake.

18. Soal 18

```
#%% soal18
    if MODE == 'train':
        for epoch in range(epochs):
            print("Epoch:", epoch)

            gen_losses = []
            dis_losses = []
```

Kode di atas befungsi untuk melakukan training epoch, karena jika epoch semakin banyak maka kualiatas training yang dihasilkan akan semakin baik.

19. Soal 19

```
#%% soal19
            number_of_batches = int(volumes.shape[0] / batch_size)
            print("Number of batches:", number_of_batches)
            for index in range(number_of_batches):
                print("Batch:", index + 1)
```

Batch adalah jumlah file yang akan di training.

20. Soal 20

```
#%% soal20
                z_sample = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
                volumes_batch = volumes[index * batch_size:(index + 1) * batch_size, :, :, :]
```

Kode di atas befungsi untuk untuk membuat gambar bersih dari noise dan juga menyesuaikan shape.

21. Soal 21

```
#%% soal21
                # Next, generate volumes using the generate network
                gen_volumes = generator.predict_on_batch(z_sample)
```

Kode di atas befungsi untuk membuat sample gambar palsu yang akan diteruskan ke diskriminator.

22. Soal 22

```
#%% soal22
                discriminator.trainable = True
                if index % 2 == 0:
                    loss_real = discriminator.train_on_batch(volumes_batch, labels_real)
                    loss_fake = discriminator.train_on_batch(gen_volumes, labels_fake)

                    d_loss = 0.5 * np.add(loss_real, loss_fake)
                    print("d_loss:{}".format(d_loss))

                else:
                    d_loss = 0.0
```

Kode di atas befungsi untuk membuat diskriminator bisa load gambar fake dan real dari generator, oleh karena itu ada generator loss dan diskriminator loss untuk melihat seberapa baik kualitas yang dihasilkan.

23. Soal 23

```
#%% soal23
                discriminator.trainable = False
                """
                Train the generator network
                """
                z = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
                g_loss = adversarial_model.train_on_batch(z, labels_real)
                print("g_loss:{}".format(g_loss))

                gen_losses.append(g_loss)
                dis_losses.append(d_loss)
```

Kode di atas befungsi untuk melakukan print gloss untuk generator dan juga dloss untuk diskriminator.

24. Soal 24

```
#%% soal24
                # Every 10th mini-batch, generate volumes and save them
                if index % 10 == 0:
                    z_sample2 = np.random.normal(0, 0.33, size=[batch_size, 1, 1, 1, z_size]).astype(np.float32)
                    generated_volumes = generator.predict(z_sample2, verbose=3)
```

```
                    for i, generated_volume in enumerate(
    generated_volumes[:5]):
                        voxels = np.squeeze(generated_volume)
                        voxels[voxels < 0.5] = 0.
                        voxels[voxels >= 0.5] = 1.
                        saveFromVoxels(voxels, "results/img_{}_{}
    _{}".format(epoch, index, i))
```

Mengapa ada perulangan ? karena untuk melakukan perbandingan dari hasil yang sudah didapat.

25. Soal 25

```
#%% soal25
            # Write losses to Tensorboard
            write_log(tensorboard, 'g_loss', np.mean(gen_losses),
     epoch)
            write_log(tensorboard, 'd_loss', np.mean(dis_losses),
     epoch)
```

TensorBoard adalah sebuah aplikasi web localhost untuk memeriksa dan menyelesaikan grafik dari hasil TensorFlow.

26. Soal 26

```
#%% soal26
        generator.save_weights(os.path.join("models", "
    generator_weights.h5"))
        discriminator.save_weights(os.path.join("models", "
    discriminator_weights.h5"))
```

File H5 adalah file data yang disimpan dalam Format Data Hirarki (HDF). Ini berisi array multidimensi data ilmiah.

27. Soal 27

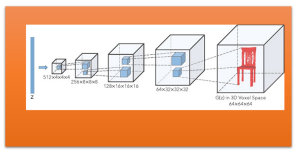
```
#%% soal27
    if MODE == 'predict':
        # Create models
        generator = build_generator()
        discriminator = build_discriminator()

        # Load model weights
        generator.load_weights(os.path.join("models", "
    generator_weights.h5"), True)
        discriminator.load_weights(os.path.join("models", "
    discriminator_weights.h5"), True)

        # Generate 3D models
        z_sample = np.random.normal(0, 1, size=[batch_size, 1, 1,
     1, z_size]).astype(np.float32)
```

```
13          generated_volumes = generator.predict(z_sample, verbose
    =3)
14
15          for i, generated_volume in enumerate(generated_volumes
    [:2]):
16              voxels = np.squeeze(generated_volume)
17              voxels[voxels < 0.5] = 0.
18              voxels[voxels >= 0.5] = 1.
19              saveFromVoxels(voxels, "results/gen_{}".format(i))
```

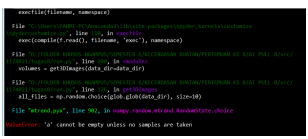
Ini adalah tahap akhir untuk melakukan testing dari model yang telah dibuat dan buat model dari yang sudah di create sebelumnya yaitu generator dan diskriminator. Untuk ilustrasi gambar sebagai berikut :



Gambar 8.12 Praktek 27

8.1.3 Penanganan Error

1. ValueError

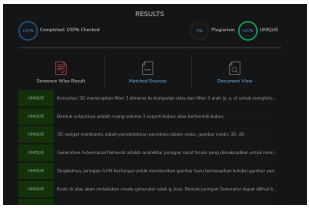


Gambar 8.13 Value error

2. Cara Penanganan Error

   - ValueError
     Error tersebut karena disebabkan gagal load dataset karena salah penamaan direktori.

8.1.4 Bukti Tidak Plagiat



Gambar 8.14 Bukti Tidak Melakukan Plagiat Chapter 8

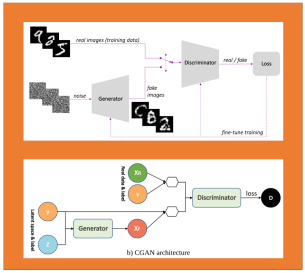
BAB 9

CHAPTER 9

9.1 1164013 - Ikrima Ningrumsari Mulyana

9.1.1 Teori

1. Jelaskan dengan ilustrasi gambar sendiri apa perbedaan antara vanilla GAN dan cGAN.
   Perbedaan antara vanilla GAN dan cGAN terdapat pada saat input proses generator, vanilla GAN memakai data noise yang di proses menjadi data fake, kalau cGAN memakai latent space atau label untuk generator. Untuk ilustrasi, lihat gambar berikut:

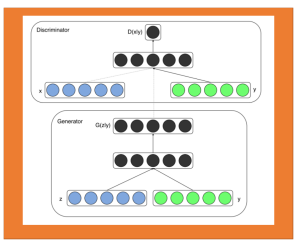


Gambar 9.1 Teori 1

2. Jelaskan dengan ilustrasi gambar sendiri arsitektur dari Age-cGAN.
   Untuk arsitektur dari Age cGAN mempunyai 4 yaitu :

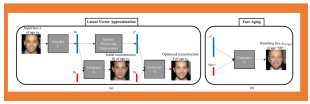
   - Encoder
   - FaceNet
   - Generator
   - Discriminator

   Untuk ilustrasi, lihat gambar berikut:



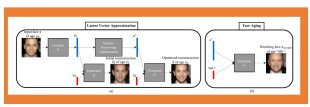
Gambar 9.2 Teori 2

3. Jelaskan dengan ilustrasi gambar sendiri arsitektur encoder network dari Agec-GAN.
   Encoder mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z. Jaringan encoder menghasilkan vektor laten dari gambar input. Jaringan Encoder adalah CNN yang mengambil gambar dari dimensi (64, 64, 3) dan mengubahnya menjadi vektor 100 dimensi. Ada empat blok konvolusional dan dua lapisan padat. Setiap blok konvolusional memiliki lapisan konvolusional, diikuti oleh lapisan normalisasi batch, dan fungsi aktivasi kecuali lapisan konvolusional pertama.



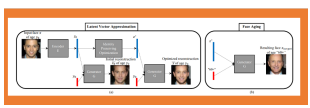
Gambar 9.3 Teori 3

4. Jelaskan dengan ilustrasi gambar sendiri arsitektur generator network dari Agec-GAN.
   Generator dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar. Generator adalah CNN dan dibutuhkan vektor laten 100 dimensi dan vektor kondisi y, dan mencoba menghasilkan gambar realistis dari dimensi (64, 64, 3). Generator memiliki lapisan padat, membingungkan, dan konvolusional. Dibutuhkan dua input satu adalah vektor noise dan yang kedua adalah vektor kondisi. Vektor kondisi adalah informasi tambahan yang disediakan untuk jaringan. Untuk Age-cGAN, ini akan menjadi age.



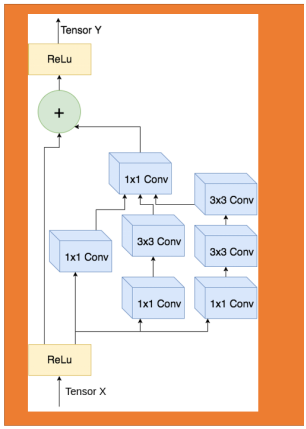
Gambar 9.4 Teori 4

5. Jelaskan dengan ilustrasi gambar sendiri arsitektur discriminator network dari Age-cGAN.
   Diskriminator mencoba membedakan antara gambar asli dan gambar palsu. Diskriminator adalah CNN dan memprediksi gambar yang diberikan adalah nyata atau palsu. Ada beberapa blok konvolusional. Setiap blok konvolusional berisi lapisan konvolusional yang diikuti oleh lapisan normalisasi batch, dan fungsi aktivasi, kecuali blok konvolusional pertama, yang tidak memiliki lapisan normalisasi batch.



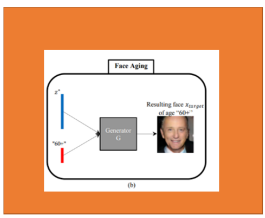
Gambar 9.5 Teori 5

6. Jelaskan dengan ilustrasi gambar apa itu pretrained Inception-ResNet-2 Model.
   Model Inception-ResNet v2 adalah model yang ciptakan untuk keperluan klasifikasi image dengan bobot di ImageNet.



Gambar 9.6 Teori 6

7. Jelaskan dengan ilustrasi gambar sendiri arsitektur Face recognition network Age-cGAN.
   FaceNet: Ini adalah jaringan pengenalan wajah yang mempelajari perbedaan antara gambar input x dan gambar yang direkonstruksi x. FaceNet mengenali identitas seseorang dalam gambar yang diberikan. Model Inception, ResNet-50 atau Inception-ResNet-2 yang telah dilatih sebelumnya tanpa lapisan yang terhubung sepenuhnya dapat digunakan. Embedding yang diekstraksi untuk gambar asli dan gambar yang direkonstruksi dapat dihitung dengan menghitung jarak Euclidean dari embeddings.

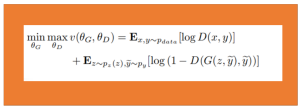


Gambar 9.7 Teori 7

8. Sebutkan dan jelaskan serta di sertai contoh-contoh tahapan dari Age-cGAN.
   Untuk tahapan dari Age cGAN yaitu :

   - Input
   - Training
   - Testing

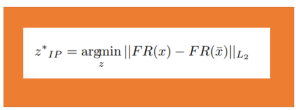
9. Berikan contoh perhitungan fungsi training objektif
   Untuk penjelasan tersebut dijelaskan pada gambar dibawah ini : Fungsi obyektif training untuk training cGAN Dimana log D (x, y) adalah kerugian untuk model Diskriminator. log (1-D (G (x, y '), y')) adalah kerugian untuk model Generator. P (data) adalah distribusi dari semua gambar yang mungkin.



Gambar 9.8 Teori 9

10. Berikan contoh dengan ilustrasi penjelasan dari Initial latent vector approximation.
    Initial latent vector approximation: Encoder network training adakah seubah metode perkiraan awal vektor laten digunakan untuk memperkirakan vektor laten untuk mengoptimalkan rekonstruksi gambar wajah. Encoder adalah jaringan saraf yang mendekati vektor laten. Kami melatih jaringan encoder pada gambar yang dihasilkan dan gambar nyata. Setelah dilatih, jaringan encoder akan mulai menghasilkan vektor laten dari distribusi yang dipelajari. Fungsi tujuan pelatihan untuk melatih jaringan encoder adalah kehilangan jarak Euclidean.

11. Berikan contoh perhitungan latent vector optimization



Gambar 9.9 Teori 11

9.1.2 Praktek Program

1. Jelaskan bagaimana cara ekstrak file dataset Age-cGAN menggunakan google colab.

```
from google.colab import drive
drive.mount('/content/drive')

import tarfile
tf = tarfile.open("/content/drive/My Drive/Colab Notebooks/GIT CH
    9/data/wiki_crop.tar")
tf.extractall(path="/content/drive/My Drive/Colab Notebooks/GIT
    CH 9/data")
```

Kode di atas akan melakukan mount dan extract dataset.

- Login ke google colab menggunakan akun google
- Mount google drive
- Lakukan proses unzip melalui notebook python di google colab, unzip pakai codingan
- Selesai

2. Jelaskan bagaimana kode program bekerja untuk melakukan load terhadap dataset yang sudah di ekstrak, termasuk bagaimana penjelasan kode program perhitungan usia.

```
def calculate_age(taken, dob):
    birth = datetime.fromordinal(max(int(dob) - 366, 1))

    if birth.month < 7:
        return taken - birth.year
    else:
        return taken - birth.year - 1

#%%
def load_data(wiki_dir, dataset='wiki'):
    # Load the wiki.mat file
    meta = loadmat(os.path.join(wiki_dir, "{}.mat".format(dataset
    )))

    # Load the list of all files
    full_path = meta[dataset][0, 0]["full_path"][0]

    # List of Matlab serial date numbers
    dob = meta[dataset][0, 0]["dob"][0]

    # List of years when photo was taken
    photo_taken = meta[dataset][0, 0]["photo_taken"][0]  # year

    # Calculate age for all dobs
    age = [calculate_age(photo_taken[i], dob[i]) for i in range(
    len(dob))]

    # Create a list of tuples containing a pair of an image path
    and age
    images = []
    age_list = []
    for index, image_path in enumerate(full_path):
        images.append(image_path[0])
        age_list.append(age[index])
```

Kode di atas untuk load data dan melakukan fungsi perhitungan usia.

3. Jelaskan bagaimana kode program The Encoder Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana.

```
def build_encoder():
```

```
    """
    Encoder Network
    """
    input_layer = Input(shape=(64, 64, 3))

    # 1st Convolutional Block
    enc = Conv2D(filters=32, kernel_size=5, strides=2, padding='
    same')(input_layer)
    # enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

    # 2nd Convolutional Block
    enc = Conv2D(filters=64, kernel_size=5, strides=2, padding='
    same')(enc)
    enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

    # 3rd Convolutional Block
    enc = Conv2D(filters=128, kernel_size=5, strides=2, padding='
    same')(enc)
    enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

    # 4th Convolutional Block
    enc = Conv2D(filters=256, kernel_size=5, strides=2, padding='
    same')(enc)
    enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

    # Flatten layer
    enc = Flatten()(enc)

    # 1st Fully Connected Layer
    enc = Dense(4096)(enc)
    enc = BatchNormalization()(enc)
    enc = LeakyReLU(alpha=0.2)(enc)

    # Second Fully Connected Layer
    enc = Dense(100)(enc)

    # Create a model
    model = Model(inputs=[input_layer], outputs=[enc])
    return model
```

Encoder berfungsi untuk mempelajari pemetaan terbalik dari gambar wajah input dan kondisi usia dengan vektor laten Z.

4. Jelaskan bagaimana kode program The Generator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana.

```
def build_generator():
    """
    Create a Generator Model with hyperparameters values defined
    as follows
    """
```

```
    latent_dims = 100
    num_classes = 6

    input_z_noise = Input(shape=(latent_dims,))
    input_label = Input(shape=(num_classes,))

    x = concatenate([input_z_noise, input_label])

    x = Dense(2048, input_dim=latent_dims + num_classes)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dropout(0.2)(x)

    x = Dense(256 * 8 * 8)(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dropout(0.2)(x)

    x = Reshape((8, 8, 256))(x)

    x = UpSampling2D(size=(2, 2))(x)
    x = Conv2D(filters=128, kernel_size=5, padding='same')(x)
    x = BatchNormalization(momentum=0.8)(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = UpSampling2D(size=(2, 2))(x)
    x = Conv2D(filters=64, kernel_size=5, padding='same')(x)
    x = BatchNormalization(momentum=0.8)(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = UpSampling2D(size=(2, 2))(x)
    x = Conv2D(filters=3, kernel_size=5, padding='same')(x)
    x = Activation('tanh')(x)

    model = Model(inputs=[input_z_noise, input_label], outputs=[x
    ])
    return model
```

Generator network agar bekerja dengan baik dibutuhkan representasi tersembunyi dari gambar wajah dan vektor kondisi sebagai input dan menghasilkan gambar.

5. Jelaskan bagaimana kode program The Discriminator Network bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana.

```
def build_discriminator():
    """
    Create a Discriminator Model with hyperparameters values
    defined as follows
    """
    input_shape = (64, 64, 3)
    label_shape = (6,)
    image_input = Input(shape=input_shape)
    label_input = Input(shape=label_shape)

```

```
    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(
    image_input)
    x = LeakyReLU(alpha=0.2)(x)

    label_input1 = Lambda(expand_label_input)(label_input)
    x = concatenate([x, label_input1], axis=3)

    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Conv2D(256, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Conv2D(512, kernel_size=3, strides=2, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.2)(x)

    x = Flatten()(x)
    x = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=[image_input, label_input], outputs=[x])
    return model
```

Diskriminator mencoba untuk membedakan antara gambar asli dan gambar palsu.

6. Jelaskan bagaimana kode program Training cGAN bekerja dijelaskan dengan bahasa awam dengan ilustrasi sederhana.

```
if __name__ == '__main__':
    # Define hyperparameters
    data_dir = "data"
    wiki_dir = os.path.join(data_dir, "wiki_crop1")
    epochs = 500
    batch_size = 2
    image_shape = (64, 64, 3)
    z_shape = 100
    TRAIN_GAN = True
    TRAIN_ENCODER = False
    TRAIN_GAN_WITH_FR = False
    fr_image_shape = (192, 192, 3)

    # Define optimizers
    dis_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
    epsilon=10e-8)
    gen_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2=0.999,
    epsilon=10e-8)
    adversarial_optimizer = Adam(lr=0.0002, beta_1=0.5, beta_2
    =0.999, epsilon=10e-8)

    """
    Build and compile networks
    """
```

```
    # Build and compile the discriminator network
    discriminator = build_discriminator()
    discriminator.compile(loss=['binary_crossentropy'], optimizer
    =dis_optimizer)

    # Build and compile the generator network
    generator = build_generator()
    generator.compile(loss=['binary_crossentropy'], optimizer=
    gen_optimizer)

    # Build and compile the adversarial model
    discriminator.trainable = False
    input_z_noise = Input(shape=(100,))
    input_label = Input(shape=(6,))
    recons_images = generator([input_z_noise, input_label])
    valid = discriminator([recons_images, input_label])
    adversarial_model = Model(inputs=[input_z_noise, input_label
    ], outputs=[valid])
    adversarial_model.compile(loss=['binary_crossentropy'],
    optimizer=gen_optimizer)

    tensorboard = TensorBoard(log_dir="logs/{}".format(time.time
    ()))
    tensorboard.set_model(generator)
    tensorboard.set_model(discriminator)

    """
    Load the dataset
    """
    images, age_list = load_data(wiki_dir=wiki_dir, dataset="wiki
    ")
    age_cat = age_to_category(age_list)
    final_age_cat = np.reshape(np.array(age_cat), [len(age_cat),
    1])
    classes = len(set(age_cat))
    y = to_categorical(final_age_cat, num_classes=len(set(age_cat
    )))

    loaded_images = load_images(wiki_dir, images, (image_shape
    [0], image_shape[1]))

    # Implement label smoothing
    real_labels = np.ones((batch_size, 1), dtype=np.float32) *
    0.9
    fake_labels = np.zeros((batch_size, 1), dtype=np.float32) *
    0.1

    """
    Train the generator and the discriminator network
    """
    if TRAIN_GAN:
        for epoch in range(epochs):
            print("Epoch:{}".format(epoch))

            gen_losses = []
            dis_losses = []
```

```
67
68          number_of_batches = int(len(loaded_images) /
    batch_size)
69          print("Number of batches:", number_of_batches)
70          for index in range(number_of_batches):
71              print("Batch:{}".format(index + 1))
72
73              images_batch = loaded_images[index * batch_size:(
    index + 1) * batch_size]
74              images_batch = images_batch / 127.5 - 1.0
75              images_batch = images_batch.astype(np.float32)
76
77              y_batch = y[index * batch_size:(index + 1) *
    batch_size]
78              z_noise = np.random.normal(0, 1, size=(batch_size
    , z_shape))
79
80              """
81              Train the discriminator network
82              """
83
84              # Generate fake images
85              initial_recon_images = generator.predict_on_batch
    ([z_noise, y_batch])
86
87              d_loss_real = discriminator.train_on_batch([
    images_batch, y_batch], real_labels)
88              d_loss_fake = discriminator.train_on_batch([
    initial_recon_images, y_batch], fake_labels)
89
90              d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
91              print("d_loss:{}".format(d_loss))
92
93              """
94              Train the generator network
95              """
96
97              z_noise2 = np.random.normal(0, 1, size=(
    batch_size, z_shape))
98              random_labels = np.random.randint(0, 6,
    batch_size).reshape(-1, 1)
99              random_labels = to_categorical(random_labels, 6)
100
101             g_loss = adversarial_model.train_on_batch([
    z_noise2, random_labels], [1] * batch_size)
102
103             print("g_loss:{}".format(g_loss))
104
105             gen_losses.append(g_loss)
106             dis_losses.append(d_loss)
107
108         # Write losses to Tensorboard
109         write_log(tensorboard, 'g_loss', np.mean(gen_losses),
     epoch)
110         write_log(tensorboard, 'd_loss', np.mean(dis_losses),
     epoch)
```

```
            """
            Generate images after every 10th epoch
            """
            if epoch % 10 == 0:
                images_batch = loaded_images[0:batch_size]
                images_batch = images_batch / 127.5 - 1.0
                images_batch = images_batch.astype(np.float32)

                y_batch = y[0:batch_size]
                z_noise = np.random.normal(0, 1, size=(batch_size, z_shape))

                gen_images = generator.predict_on_batch([z_noise, y_batch])

                for i, img in enumerate(gen_images[:5]):
                    save_rgb_img(img, path="results/img_{}_{}.png".format(epoch, i))

        # Save networks
        try:
            generator.save_weights("generator.h5")
            discriminator.save_weights("discriminator.h5")
        except Exception as e:
            print("Error:", e)
```

Proses training dengan load file .mat pada dataset, lalu epoch sebanyak 500 kali.

7. Jelaskan bagaimana kode program Initial dan latent vector approximation bekerja dijelaskan dengan bahawa awam dengan ilustrasi sederhana.

```
if TRAIN_ENCODER:
    # Build and compile encoder
    encoder = build_encoder()
    encoder.compile(loss=euclidean_distance_loss, optimizer='adam')

    # Load the generator network's weights
    try:
        generator.load_weights("generator.h5")
    except Exception as e:
        print("Error:", e)

    z_i = np.random.normal(0, 1, size=(5000, z_shape))

    y = np.random.randint(low=0, high=6, size=(5000,), dtype=np.int64)
    num_classes = len(set(y))
    y = np.reshape(np.array(y), [len(y), 1])
    y = to_categorical(y, num_classes=num_classes)

    for epoch in range(epochs):
        print("Epoch:", epoch)

```

```
22              encoder_losses = []
23
24              number_of_batches = int(z_i.shape[0] / batch_size)
25              print("Number of batches:", number_of_batches)
26              for index in range(number_of_batches):
27                  print("Batch:", index + 1)
28
29                  z_batch = z_i[index * batch_size:(index + 1) *
    batch_size]
30                  y_batch = y[index * batch_size:(index + 1) *
    batch_size]
31
32                  generated_images = generator.predict_on_batch([
    z_batch, y_batch])
33
34                  # Train the encoder model
35                  encoder_loss = encoder.train_on_batch(
    generated_images, z_batch)
36                  print("Encoder loss:", encoder_loss)
37
38                  encoder_losses.append(encoder_loss)
39
40              # Write the encoder loss to Tensorboard
41              write_log(tensorboard, "encoder_loss", np.mean(
    encoder_losses), epoch)
42
43          # Save the encoder model
44          encoder.save_weights("encoder.h5")
45
46      """
47      Optimize the encoder and the generator network
48      """
49      if TRAIN_GAN_WITH_FR:
50
51          # Load the encoder network
52          encoder = build_encoder()
53          encoder.load_weights("encoder.h5")
54
55          # Load the generator network
56          generator.load_weights("generator.h5")
57
58          image_resizer = build_image_resizer()
59          image_resizer.compile(loss=['binary_crossentropy'],
    optimizer='adam')
60
61          # Face recognition model
62          fr_model = build_fr_model(input_shape=fr_image_shape)
63          fr_model.compile(loss=['binary_crossentropy'], optimizer=
    "adam")
64
65          # Make the face recognition network as non-trainable
66          fr_model.trainable = False
67
68          # Input layers
69          input_image = Input(shape=(64, 64, 3))
70          input_label = Input(shape=(6,))
```

```
71
72          # Use the encoder and the generator network
73          latent0 = encoder(input_image)
74          gen_images = generator([latent0, input_label])
75
76          # Resize images to the desired shape
77          resized_images = Lambda(lambda x: K.resize_images(
    gen_images, height_factor=3, width_factor=3,
78
    data_format='channels_last'))(gen_images)
79          embeddings = fr_model(resized_images)
80
81          # Create a Keras model and specify the inputs and outputs
     for the network
82          fr_adversarial_model = Model(inputs=[input_image,
    input_label], outputs=[embeddings])
83
84          # Compile the model
85          fr_adversarial_model.compile(loss=euclidean_distance_loss
    , optimizer=adversarial_optimizer)
86
87          for epoch in range(epochs):
88              print("Epoch:", epoch)
89
90              reconstruction_losses = []
91
92              number_of_batches = int(len(loaded_images) /
    batch_size)
93              print("Number of batches:", number_of_batches)
94              for index in range(number_of_batches):
95                  print("Batch:", index + 1)
96
97                  images_batch = loaded_images[index * batch_size:(
    index + 1) * batch_size]
98                  images_batch = images_batch / 127.5 - 1.0
99                  images_batch = images_batch.astype(np.float32)
100
101                 y_batch = y[index * batch_size:(index + 1) *
    batch_size]
102
103                 images_batch_resized = image_resizer.
    predict_on_batch(images_batch)
104
105                 real_embeddings = fr_model.predict_on_batch(
    images_batch_resized)
106
107                 reconstruction_loss = fr_adversarial_model.
    train_on_batch([images_batch, y_batch], real_embeddings)
108
109                 print("Reconstruction loss:", reconstruction_loss
    )
110
111                 reconstruction_losses.append(reconstruction_loss)
112
113             # Write the reconstruction loss to Tensorboard
```

```
            write_log(tensorboard, "reconstruction_loss", np.mean
    (reconstruction_losses), epoch)

            """
            Generate images
            """
            if epoch % 10 == 0:
                images_batch = loaded_images[0:batch_size]
                images_batch = images_batch / 127.5 - 1.0
                images_batch = images_batch.astype(np.float32)

                y_batch = y[0:batch_size]
                z_noise = np.random.normal(0, 1, size=(batch_size
    , z_shape))

                gen_images = generator.predict_on_batch([z_noise,
     y_batch])

                for i, img in enumerate(gen_images[:5]):
                    save_rgb_img(img, path="results/img_opt_{}_
    {}.png".format(epoch, i))

        # Save improved weights for both of the networks
        generator.save_weights("generator_optimized.h5")
        encoder.save_weights("encoder_optimized.h5")
```

Proses kerja nya dengan membuat model .h5, lalu load data dengan menghasilkan result.

9.1.3 Penanganan Error

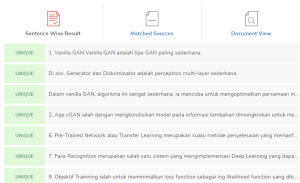
1. ValueError

Gambar 9.10 FileNotFoundError

2. Cara Penanganan Error

   - FileNotFoundError
     Error tersebut karena disebabkan gagal load dataset karena salah penamaan type file.

9.1.4 Bukti Tidak Plagiat



Gambar 9.11 Bukti Tidak Melakukan Plagiat Chapter 9

BAB 10

CHAPTER 10

BAB 11

CHAPTER 11

BAB 12

CHAPTER 12

BAB 13

CHAPTER 13

BAB 14

CHAPTER 14

DAFTAR PUSTAKA

[1] R. Awangga, “Sampeu: Servicing web map tile service over web map service to increase computation performance,” in *IOP Conference Series: Earth and Environmental Science*, vol. 145, no. 1. IOP Publishing, 2018, p. 012057.

Index